

# Testovi stresa i opterećenja

U okviru ove lekcije govorićemo o dve tehnike testiranja crne kutije: stress i load testiranju. Upoznaćemo se sa ovim konceptima, bolje ćemo razumeti njihovu svrhu u softverskom testiranju i videti nekoliko metoda za njihovu realizaciju.

Stres testiranje (stress testing) predstavlja test performansi programa i podrazumeva test stabilnosti sistema u situacijama izvan normalnog operativnog kapaciteta. Ima za cilj da se provere performanse sistema u ekstremnim uslovima rada i da se na taj način utvrde granice izdržljivosti sistema. Sa određenim granicama izdržljivosti znamo u kom trenutku možemo očekivati pad aplikacije, pojavu grešaka i slično.

Ovi testovi se često sprovode sa izuzetno velikim brojem korisnika i sa ogromnom količinom podataka i na ovaj način se mogu proveriti opterećenja kod velikog broja posetilaca na nekom web serveru ili simulirati napadi uskraćivanjem usluge preko raznih skripti, web robota i dr.

Sam termin *stres testiranje* koristi se kod testiranja raznih hardverskih komponenata radi provere izdržljivosti i često u bankarskom poslovanju radi provere finansijske institucije u rizičnim i teškim uslovima poslovanja. Stres test je kao termin nastao kod testiranja materijala, gde se tražila tačka pucanja, dakle tačna vrednost sile kada materijal izgubi svoju strukturu i pukne.

Poenta stres testa programa je da pronademo upravo tačku pucanja u svojim programima. Dakle, stopu opterećenja pri kojoj aplikacija prestaje da odgovara korisnicima ili počne nepravilno da radi.

Tada imamo otvorena pitanja:

- Koja je tačno tačka pucanja?
- Koliko se grešaka javlja i koliko često?
- Da li je aplikacija prestala sa radom?
- Ukoliko je prestala sa radom, da li se vratila u normalan rad?
- Da li je došlo do većeg utroška memorije servera od očekivanog?

U slučaju softvera, najčešće web aplikacija, primeri situacija u kojima aplikacija može biti pod stresom mogu biti:

1. U slučaju web sajtova sa integrisanom prodavnicom, do stresa može doći kada kompanija nudi neki vid rasprodaje/akcije; u ovim slučajevima broj poseta sajtu i broj porudžbina raste i aplikacija može imati problem zbog povećanog stresa.

Primer ovoga je nedavno bila web prodavnica kompanije Debenhams: ubrzo nakon aktiviranja rasprodaje na njenom sajtu, on je pao. Razlog je bio veliki broj neočekivanih kupaca, kako pre toga nije rađen stres test da se utvrdi koliko kupaca prodavnica može da podrži; sajt je bio trajno nedostupan za sve korisnike.

Programeri iza sajta su morali po hitnom postupku prvo da na neki način obaveste korisnike koji je razlog zašto je sajt nedostupan i ubrzo su korisnici dobijali poruku koju možete videti na narednoj slici.



**We're sorry - our site's not available right now.**

We're experiencing large volumes of traffic, but don't worry we'll be back soon.

*Slika 12.1. Poruka na sajtu Debenhams*

Na ovaj način su mogli da obaveste korisnike da je sajt pao zbog prevelikog saobraćaja. Takođe, morali su hitno da naprave ispravke na kodu tako da se prodaja nastavi, i jedino brzo rešenje je bilo da se uvede queue sistem – sistem čekanja u redu na pristup sajtu. Kada je ovaj sistem uveden, u jednom trenutku je zabeleženo 900.000 korisnika koji su čekali na ulaz u web prodavnicu. Progameri su takođe morali da ograniče vreme zadržavanja na sajtu na 30 minuta.

2. Čest slučaj ovakih problema na manjem nivou se dešava i sa blog stranicama. Blog sajtovi se uvek kreiraju za manji saobraćaj, ali neočekivan skok saobraćaja može biti u slučaju npr. objavljivanja bloga u nekim vestima. Ovaj tip problema, naravno, ne važi samo za male blogove. Nakon vesti o smrti popularnog pevača Michaela Jacksona, veliki broj satova koji su cirkulisali ove vesti je pao usled prevelikog saobraćaja. Google je imao problem sa prikazivanjem sadržaja u vezi sa ovim vestima. Wikipedia je preventivno isključila pristup Wiki stranici o pevaču; CNN i BBC su imali problem u radu svojih sajtova itd.

Ovo su samo neki od primera koji ilustruju apsolutnu važnost dobrog stres testa aplikacije. Svi sistemi čija je svrha da umanje broj korisnika i smanje zadržavanje korisnika na sajtu su uvedeni hitno nakon pojave greške. Ovako nešto nije dobro, jer utiče na reputaciju kompanije i smanjuje prodaju. Obavljanje stres testa pre nego što aplikacija uđe u upotrebu pomaže da odredimo koji broj korisnika aplikacija podržava i da kreiramo sistem upravo za obaveštavanje korisnika i, ukoliko je već potrebno, virtuelnog čekanja u redu na pristup.

Load testiranje je identično stres testiranju, s tim što, kada pronađemo tačku pucanja aplikacije, load testom određujemo koja je to najviša vrednost broja korisnika ili upotrebe pod kojom aplikacija može stabilno raditi u kontinuitetu, bez pojave grešaka. Dakle, poenta load testa nije maksimalno opteretiti aplikaciju, već pronaći neke idealne granice rada aplikacije.

### **Kako obavljam testiranje aplikacije?**

Testiranje aplikacije se obavlja u koracima:

1. Planiranje testa – U okviru ove faze prikupljamo podatke o sistemu, analiziramo ga i definišemo ciljeve. Dakle, dajemo pretpostavke o broju korisnika, vremenu koje provode u korišćenju aplikacije itd.
2. Definisanje skripte za testiranje – U ovoj fazi kreiramo skriptu koja će pristupati aplikaciji i vršiti pritisak na njen rad.
3. Izvršavanje skripte za testiranje – U okviru ove faze pratimo rad skripte i beležimo rezultate rada aplikacije, kao i pojavu grešaka, usporavanje rada itd.
4. Analiza rezultata – Analizom dobijenih podataka pokušavamo da utvrdimo u kom delu aplikacije ili servera nastaje problem.
5. Podešavanje i optimizacija – U poslednjoj fazi unosimo korekcije u kod, vršimo optimizaciju koda, pa čak i prenos saobraćaja na neki drugi deo koda.

Nakon ovih koraka, ponavljamo postupak i pratimo razlike u ponašanju programa, da bismo verifikovali da su promene donele unapređenje rada. Ovaj postupak se može ponavljati neograničen broj puta sve dok nismo zadovoljni radom programa prema definisanom opterećenju.

## Alati za testiranje aplikacije

Sada se možda pitate – koji alati ovo omogućuju i kako se generalno može izvršiti testiranje stresa? Alati ove grupe imaju zadatak da simuliraju prave korisnike i njihove zahteve od programa. Naravno, potreba za ovim alatima je bila neizbežna jer je nemoguće koordinisati testiranje programa gde je potrebno na desetine, a nekada i stotine hiljada korisnika koji istovremeno koriste web aplikaciju.

Stoga su i nastali alati koji simuliraju korisnika. Pomoću ovih aplikacija možemo postići simulaciju korisnika, od toga da samo pristupaju aplikaciji do toga da unose podatke u neku formu i šalju zahteve za podacima. Na ovaj način, aplikaciju namenjenu za 1000 istovremenih korisnika možemo da opteretimo sa 2000 i pratimo ponašanje programa kada pokušava da odgovori korisnicima.

U okviru ove lekcije govorićemo o osnovnom programu koji nam pomaže da obavimo stres testiranje nekog web sajta ili web aplikacije. To je Apache Bench.


### Apache Bench

Apache Bench (ab) predstavlja benchmark alat, tj. alat za merenje performansi. Apache Bench koristimo za merenje performansi servera kroz HTTP protokol. Ovaj jednostavni alat dizajniran je tako da nam vrlo brzo pruži informacije koje su nam potrebne da procenimo koliko dobro naša aplikacija radi. Takođe, dobijamo vrednu informaciju o tome koliko klijenata naša aplikacija može da usluži u trenutku vremena.

Pogledajmo kako instaliramo i koristimo ovaj alat. Pre svega, potrebno je da preuzmemo instalaciju ovog programa. Instalaciju najnovije verzije softvera možete pronaći na sledećem linku:

<https://www.apachelounge.com/download/>

Kada pristupimo sajtu, u sekciji *Apache binaries* je potrebno odabrati verziju za vaš operativni sistem (slika 12.2).



# Apache Lounge

Webmasters

- Home
- VS16**
- VC15
- Additional

## Apache 2.4 VS16 Windows Binaries and Modules

Apache Lounge has provided up-to-date Windows binaries and popular third-party modules for more than 15 years. We have hundreds of thousands of satisfied users: small and big companies as well as home users. Always build with up to date dependencies and latest compilers, and tested thorough. The binaries are referenced by the ASF, Microsoft, PHP etc. and more and more software is packaged with our binaries and modules.

The binaries, are build with the sources from ASF at [httpd.apache.org](http://httpd.apache.org), contains the latest patches and latest dependencies like zlib, openssl etc. which makes the downloads here mostly more actual then downloads from other places. The binaries **do not run** on XP and 2003. Runs on: 7 SP1, Vista SP2, 8 / 8.1, 10, Server 2008 SP2 / R2 SP1, Server 2012 / R2, Server 2016/2019.

Build with the latest Windows® Visual Studio C++ 2019 aka VS16. VS16 has improvements, fixes and optimizations over VC15 in areas like Performance, MemoryManagement, New standard conformance features, Code generation and Stability. For example code quality tuning and improvements done across different code generation areas for "speed". And makes more use of latest processors and supported Windows editions (win7 and up) internal features.


**VS16 is backward compatible**, see [Compatibility VS16](#). You can use a VC15/14 module inside a VS16 binary, for example PHP VC15/14 as module,

**Be sure** you installed latest 14.28.29913.0 Visual C++ Redistributable for Visual Studio 2015-2019 : [vc\\_redist\\_x64](#) or [vc\\_redist\\_x86](#) see [Redistributable](#)

### Apache 2.4 binaries VS16


[Info & Changelog](#)

#### Apache 2.x.x Win64


[httpd-2.4.46-win64-VS16.zip](#)
27 Mar '21 10.320k

[PGP Signature](#) (Public [PGP key](#)), SHA1-SHA512 [Checksums](#)

#### Apache 2.x.x Win32


[httpd-2.4.46-win32-VS16.zip](#)
27 Mar '21 9.442k

[PGP Signature](#) (Public [PGP key](#)), SHA1-SHA512 [Checksums](#)

#### Keep Server Online

If you find the downloads useful, please express your satisfaction with a donation.

[Donate](#)

Slika 12.2. Preuzimanje instalacije Apachea

Nakon preuzimanja zip fajla, potrebno ga je raspakovati na lokaciji koja vam odgovara; u ovom slučaju, najbolja je lokacija koja je lako dostupna kroz Command Prompt našeg operativnog sistema. Recimo, to može biti u korenu particije diska.

Kada smo kopirali Apache folder, sada nam ostaje samo da pokrenemo ab.exe unutar tog foldera. Stoga u okviru Command Prompta dolazimo do lokacije foldera Apache i pristupamo njegovom bin folderu, jer se unutar ovog foldera nalazi ab.exe fajl, koji nam je potreban da bismo pokrenuli Apache Bench.

```
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dragoljub Catovic>cd..

C:\Users>cd..

C:\>cd Apache24

C:\Apache24>cd bin

C:\Apache24\bin>
```

*Slika 12.3. Pristup bin folderu*

### **Napomena**

Sve lokacije, nazivi foldera i putanje zavise od vašeg računara; na slici 12.3. je prikazan jedan primer kako se pomoću komande `cd..` vraćamo unazad kroz foldere (jer smo sačuvali folder direktno na C particiji našeg diska). Takođe, pristupamo folderu `Apache24`, jer ovo je folder verzije sa kojom radimo u vreme pisanja kursa; u vašem slučaju postoji mogućnost da ćete koristiti noviju verziju. Stoga obavezno proverite na kojoj lokaciji ste raspakovali zip fajl, kao i nazive foldera i putanje, da biste pravilno obavili pristup folderima i fajlu kroz Command Prompt.

Kada se nalazimo u bin folderu Apache programa, ostaje samo da pokrenemo alat. To postizemo komandom `ab`; naravno, kada pokrenemo alat bez parametara, dobijamo poruku o grešakama, ali ujedno i pregled svih parametara koji su nam dostupni u okviru alata.



```

C:\Apache24\bin>ab
ab: wrong number of arguments
Usage: ab [options] [http://]hostname[:port]/path
Options are:
  -n requests      Number of requests to perform
  -c concurrency   Number of multiple requests to make at a time
  -t timelimit      Seconds to max. to spend on benchmarking
                   This implies -n 50000
  -s timeout        Seconds to max. wait for each response
                   Default is 30 seconds
  -b window size    Size of TCP send/receive buffer, in bytes
  -B address        Address to bind to when making outgoing connections
  -p postfile       File containing data to POST. Remember also to set -T
  -u putfile        File containing data to PUT. Remember also to set -T
  -T content-type   Content-type header to use for POST/PUT data, eg.
                   'application/x-www-form-urlencoded'
                   Default is 'text/plain'
  -v verbosity      How much troubleshooting info to print
  -w               Print out results in HTML tables
  -i               Use HEAD instead of GET
  -x attributes     String to insert as table attributes
  -y attributes     String to insert as tr attributes
  -z attributes     String to insert as td or th attributes
  -C attribute      Add cookie, eg. 'Apache=1234'. (repeatable)
  -H attribute      Add Arbitrary header line, eg. 'Accept-Encoding: gzip'
                   Inserted after all normal header lines. (repeatable)
  -A attribute      Add Basic WWW Authentication, the attributes
                   are a colon separated username and password.
  -P attribute      Add Basic Proxy Authentication, the attributes
                   are a colon separated username and password.
  -X proxy:port     Proxyserver and port number to use
  -V               Print version number and exit
  -k               Use HTTP KeepAlive feature
  -d               Do not show percentiles served table.
  -S               Do not show confidence estimators and warnings.
  -q               Do not show progress when doing more than 150 requests
  -l               Accept variable document length (use this for dynamic pages)
  -g filename       Output collected data to gnuplot format file.
  -e filename       Output CSV file with percentages served
  -r               Don't exit on socket receive errors.
  -m method         Method name
  -h               Display usage information (this message)

C:\Apache24\bin>

```

Slika 12.4. Lista parametara Apache Bencha

Od navedenih parametara, najviše se koriste:

Parametar	Opis funkcije
-n	broj zahteva za slanje ka web aplikaciji (svaki zahtev predstavlja jednog simuliranog korisnika)
-c	broj konkurentnih zahteva (simuliran broj korisnika koji koriste aplikaciju u isto vreme)
-t	maksimalno vreme provedeno u izvršavanju benchmarka (parametar je potreban za veliki broj korisnika)
-l	parametar koji napominje Apache Benchu da zahtevi ne moraju biti iste veličine i trajanja (koristi se za dinamičke aplikacije)
-h	ovaj parametar se koristi za prikaz kompletne liste parametara, koju možete videti na slici 12.4.

Tabela 12.1. Parametri alata Apache Bench



U redu, sada smo pokrenuli alat, uverili se da radi i objasnili koja je svrha parametara; sada želimo da pokrenemo test nad nekom aplikacijom. Recimo da želimo da testiramo kako sajt google.com obrađuje zahteve za npr. 100 korisnika, od kojih 10 u isto vreme koristi sajt. To postizemo tako što u komandnoj liniji unesemo sledeće:

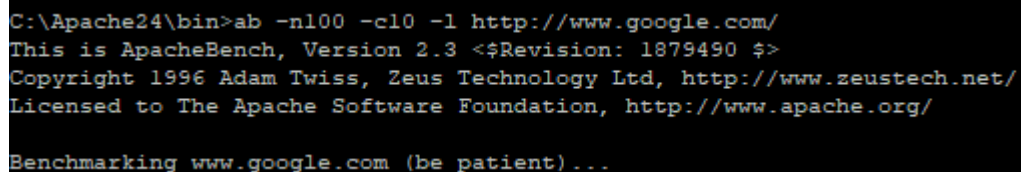
```
ab -n100 -c10 -l http://www.google.com/
```

Dakle, koristimo komandu ab da naznačimo da pozivamo Apache Bench, zatim parametar -n100; kao što smo videli u tabeli, -n govori koliko korisnika pristupa sajtu, uz naveden broj korisnika, zatim parametar -c, koji navodi koliko će istovremenih zahteva biti, sa brojem 10, dakle 10 zahteva, i potom parametar -l, koji napominje da zahtevi ne moraju biti iste veličine, što će simulirati još realniji pristup sajtu. Na samom kraju navodimo punu adresu do sajta koji želimo da testiramo po HTTP protokolu.

### Napomena

Ne možemo testirati sajtove pod HTTPS protokolom u okviru standardnog Apache Bench, jer ovaj protokol štiti sajtove od ovog alata. Ukoliko želimo da koristimo HTTP, potrebno je test pokrenuti komandom abs, umesto ab.

Kada pokrenemo test, dobijamo sledeći prikaz:

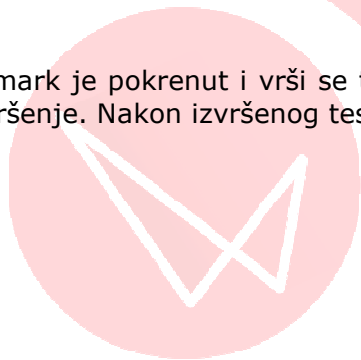


```
C:\Apache24\bin>ab -n100 -c10 -l http://www.google.com/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking www.google.com (be patient)...
```

Slika 12.5. Lista parametara Apache Bench

Dakle, benchmark je pokrenut i vrši se test stresa, i dobijamo poruku da budemo strpljivi i sačekamo izvršenje. Nakon izvršenog testa, dobijamo sledeći prikaz:



```

Server Software:      gws
Server Hostname:      www.google.com
Server Port:          80

Document Path:        /
Document Length:      Variable

Concurrency Level:    10
Time taken for tests:  79.119 seconds
Complete requests:    100
Failed requests:      0
Total transferred:    1661052 bytes
HTML transferred:     1590652 bytes
Requests per second:  1.26 [#/sec] (mean)
Time per request:     7911.945 [ms] (mean)
Time per request:     791.194 [ms] (mean, across all concurrent requests)
Transfer rate:        20.50 [Kbytes/sec] received

Connection Times (ms)
      min  mean[+/-sd] median   max
Connect:    12    779 2353.9     34   15063
Processing: 130  7020 9049.9    2257  36384
Waiting:    84  3931 4701.6    2087  18220
Total:      165  7798 9657.1    2319  36420

Percentage of the requests served within a certain time (ms)
 50%    2319
 66%    7309
 75%   14272
 80%   18235
 90%   23287
 95%   26288
 98%   36398
 99%   36420
100%   36420 (longest request)

C:\Apache24\bin>

```

*Slika 12.6. Izvršen benchmark nad sajtom google.com*

U okviru izveštaja dobijamo značajne podatke; prvenstveno možemo videti ukupno trajanje testa. Takođe, da se svih 100 zahteva izvršilo uspešno – dakle, server nije imao problem u odgovorima (što je bilo očekivano, pošto je reč o Google serverima). Takođe možemo videti da se svake sekunde slao po jedan zahtev, a možda najvažnija stavka su vremena odgovora u milisekundama, pa ćemo obratiti malo više pažnje na to:

```

Connection Times (ms)
      min  mean[+/-sd] median   max
Connect:    12    779 2353.9     34   15063
Processing: 130  7020 9049.9    2257  36384
Waiting:    84  3931 4701.6    2087  18220
Total:      165  7798 9657.1    2319  36420

```

*Slika 12.7. Trajanje obrade zahteva sajta google.com*



Connect sekcija nam govori koliko je vremena bilo potrebno da se ostvari konekcija sa sajtom.

Processing sekcija nam govori koliko vremena je bilo potrebno da aplikacija obradi zahtev.

Waiting – koliko smo ukupno čekali na odgovor.

U okviru svih prikazanih vrednosti imamo min, mean[+/-sd], median i max.

Vrednost min nam govori koje je bilo najkraće vreme realizacije, median nam govori gde je polovina vremena za svaki od parametara (u našem slučaju, 50% zahteva je bilo ispod 147 ms), max koje je bilo maksimalno vreme, dok mean[+/-sd] daje kalkulaciju proseka vremena sa odstupanjem (standardnom devijacijom), što je veoma dobar indikator performansi i stresa aplikacije i stabilnosti rada našeg servera sa aplikacijom.

U našem primeru, ukoliko se fokusiramo na vreme konektovanja, prosečno (mean) vreme je 779 ms, a standardna devijacija je 2353 ms.

Ovo znači da vreme konektovanja korisnika na serveru može biti od 0 do 3,132 (mean + sd parametar). Ovo, naravno, nije dobro, jer nam je potrebno da prosek i devijacija, tj. odstupanje budu u opsegu od 100 do 200 ms. Naravno, potrebno je imati u vidu da pristupamo testiranju Google sistema, koji ima svoj način distribucije zahteva i veliki broj servera, pa ovaj naš podatak nije jasan indikator performansi servera. Ali, važno je zapamtiti da idealan rad servera i aplikacije podrazumeva jako malu razliku između mean vrednosti i vrednosti standardne devijacije, jer to znači da naš server i aplikacija u približno istom trajanju odgovaraju svakom korisniku.

Naravno, idealno je inicijalno stres testiranje obavljati na svom računaru, dakle korišćenjem localhost adrese računara, i aplikacije samo postavljati na različitim portovima.

Pa evo, primera radi, s obzirom na to da smo instalirani Jenkins server za testiranje i on se nalazi na portu 8080, možemo testirati njegovo vreme odziva. Pošto radimo na svom računaru, možemo biti i malo agresivniji kad je reč o broju zahteva i broju istovremenih korisnika:

```
ab -n10000 -c1000 -l http://localhost:8080/
```

Dakle, sada ćemo obaviti 10.000 zahteva sa 1.000 istovremenih korisnika i ovaj saobraćaj će biti usmeren ka našem Jenkins serveru. Ukoliko sada pogledamo izveštaj:

```

C:\Apache24\bin>ab -n10000 -c1000 -l http://localhost:8080/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:      Jetty(9.4.39.v20210325)
Server Hostname:      localhost
Server Port:          8080

Document Path:        /
Document Length:       Variable

Concurrency Level:     1000
Time taken for tests:  25.877 seconds
Complete requests:     10000
Failed requests:        0
Non-2xx responses:     10000
Total transferred:     9349474 bytes
HTML transferred:      5480000 bytes
Requests per second:   386.45 [#/sec] (mean)
Time per request:      2587.687 [ms] (mean)
Time per request:      2.588 [ms] (mean, across all concurrent requests)
Transfer rate:         352.84 [Kbytes/sec] received


Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0      2   36.5        0   1035
Processing:    105 2483  813.2     2503   4211
Waiting:        0  1283  987.4     1198   4020
Total:         105 2485  813.6     2503   4211


Percentage of the requests served within a certain time (ms)
 50%    2503
 66%    2963
 75%    3028
 80%    3418
 90%    3597
 95%    3646
 98%    4030
 99%    4151
100%    4211 (longest request)

```

*Slika 12.8. Trajanje obrade zahteva localhost:8080*

Možemo videti obradu prvenstveno svakih 1000 zahteva, vreme testa i odlične rezultate. Naravno, test je objavljen na stranici koja samo prikazuje login formu, dakle login u Jenkins server, stoga su svi rezultati brži; pored toga, nismo koristili internet konekciju za pristup udaljenom serveru, već smo pristupali serveru koji je na našem računaru, pa je stoga i brzina bolja.

Naravno, i dalje možemo oboriti uslugu na portu 8080; ukoliko postavimo npr. 100.000 zahteva sa 10.000 istovremenih korisnika, odmah nakon početka testa, localhost će prestati sa radom i simulirano smo oborili uslugu na localhostu i dobili bismo poruku poput ove:



## This page isn't working

localhost is currently unable to handle this request.

HTTP ERROR 500

Reload

*Slika 12.9. Greška u odzivu adrese localhost:8080*

Dakle, data nam je povratna informacija da server ne može da obradi naš zahtev, tj. prešli smo tačku pucanja aplikacije. Ovim pristupom bismo testirali našu aplikaciju: ukoliko znamo da ne možemo da podržimo 10.000 zahteva, postepeno smanjujemo broj korisnika i pratimo brzine dok ne dođemo do zadovoljavajućih brzina i stabilnosti rada programa. Kada to postignemo, znamo koji je optimalan broj korisnika na sajtu i koliko zahteva istovremeno aplikacija može da podrži.

Naravno, ovako velike cifre nemojte ni pokušavati, jer već znamo da naši računari i njihov hardver ne mogu da podrže taj saobraćaj, jer su nam za tu količinu saobraćaja potrebne mnogo bolje serverske mašine; ovo smo uradili samo radi primera, da biste videli šta možete da očekujete kada aplikacija prestane sa radom.

### **Važna napomena**

Kao što možete da pretpostavite, stres testovi mogu napraviti problem za infrastrukturu koja podržava sajt / web aplikaciju. Kao što ste mogli da vidite, mi smo usporili rad našeg lokalnog simuliranog servera radi primera.

**Ovako nešto ne treba nikako pokušavati** sa sajtovima koji su dostupni na internetu. Sprovođenje ovih testova na serverima i sajtovima za koje nemate dozvolu za sprovođenje testa se može smatrati DoS (Denial of Service) napadom; većina ozbiljnih sajtova će vas sprečiti u ovome jer imaju mere zaštite od DoS napada, ali će ujedno i blokirati vašu IP adresu na određeno vreme, što znači da ćete izgubiti pristup sajtu čak i kada hoćete samo da ga posetite u lične svrhe. Moguće su posledice i kod vašeg internet provajdera i slično, s obzirom na to da su ovo nepoželjne radnje na internetu.

### Stres testovi se izvode isključivo:

- **na serveru koji vi posedujete (fizička serverska mašina na vašoj mreži);**
- **na vašem računaru (testiranje kroz localhost);**
- **uz dozvolu hostinga kod kog ste zakupili paket da testirate svoju aplikaciju.**

Za primer u kursu smo koristili google.com, ali sa jako malim brojem korisnika, baš iz ovih razloga, stoga je važno da vodite računa prema čemu usmeravate saobraćaj i u kojem obimu.

U okviru narednih lekcija govorićemo i o programu Apache Jmeter, koji predstavlja jedan od industrijskih standarda za stress testiranje. Ovaj alat omogućuje automatizaciju testiranja, a ujedno i kreiranje skripti za testove.

### Pitanje

Stress i load testovi su potpuno identični.

- Tačno.
- **Netačno.**

### Objašnjenje:

*Load testiranje je identično stres testiranju po pitanju realizacije, ali razlika je u tome što, kada pronađemo tačku pucanja aplikacija stress testom, load testom određujemo koja je to najviša vrednost broja korisnika ili upotrebe pod kojom aplikacija može stabilno raditi u kontinuitetu, bez pojave grešaka. Dakle, poenta load testa nije da sruši aplikaciju nego da pronađe maksimalni broj korisnika u stabilnom režimu rada.*

### Rezime

- Stres testiranje (stress testing) predstavlja test performansi programa i podrazumeva test stabilnosti sistema u situacijama izvan normalnog operativnog kapaciteta. Ima za cilj da se provere performanse sistema u ekstremnim uslovima rada i da se na taj način utvrde granice izdržljivosti sistema.
- Stres testovi se sprovode sa izuzetno velikim brojem korisnika i sa ogromnom količinom podataka i na ovaj način se mogu proveriti opterećenja kod velikog broja posetilaca na nekom web serveru ili simulacije napada uskraćivanjem usluge preko raznih skripti, web robota i dr.
- Alati za stress testiranje imaju zadatak da simuliraju prave korisnike i njihove zahteve od programa. Naravno, potreba za ovim alatima je bila neizbežna jer je nemoguće koordinisati testiranje programa gde je potrebno na desetine, a nekada i stotine hiljada korisnika koji istovremeno koriste web aplikaciju.
- Apache Bench (ab) predstavlja benchmark alat ili alat za merenje performansi. Apache Bench koristimo za merenje performansi servera kroz HTTP protocol. Ovaj jednostavan alat je dizajniran tako da nam vrlo brzo pruži informacije koje su nam potrebne da procenimo koliko dobro naša aplikacija radi.
- Load testiranje je identično stress testiranju po pitanju realizacije, a razlika je u tome što, kada pronađemo tačku pucanja aplikacija stress testom, load testom određujemo koja je to najviša vrednost broja korisnika ili upotrebe pod kojom aplikacija može stabilno raditi u kontinuitetu, bez pojave grešaka.